Defining new argumentation-based semantics by minimal models

Juan Carlos Nieves and Ulises Cortés
Universitat Politècnica de Catalunya
Software Department
c/Jordi Girona 1-3, E08034, Barcelona, Spain
{jcnieves,ia}@lsi.upc.edu

Mauricio Osorio
Universidad de las Américas - Puebla
CENTIA, Sta. Catarina Mártir
Cholula, Puebla, 72820 México
osoriomauri@googlemail.com

Ivan Olmos and Jesus A. Gonzalez
Instituto Nacional de Astrofisica, Optica y Electronica
Luis Enrique Erro No. 1, Sta Maria Tonantzintla, México, C.P. 72840
{iolmos,jagonzalez}@inaoep.mx

Abstract

Dung's argumentation approach is a unifying approach which has played an influential role on argumentation research and Artificial Intelligence (AI). Based on a proper representation of Dung's argumentation approach and minimal models, we introduce a novel argumentation semantics called preferred$^+$ semantics which follows the preferred semantics' philosophy. Also, we show how to infer preferred$^+$ semantics using a software tool called SI-COBRA that was introduced recently.

Key words *Argumentation, Logic programming, subgraph isomorphism.*

1. Introduction

The main purpose of argumentation theory is to study the fundamental mechanism, humans use in argumentation, and to explore ways to implement this mechanism on computers. In fact, argumentation theory has a wider range of application, for instance, argumentation is gaining increasing importance as a fundamental approach in multi-agent interaction, mainly because it enables rational dialogue and because it enables richer forms of negotiation that have hitherto been possible in game theory or heuristic based models [21]. Also, argumentation theory is a suitable approach for practical and uncertain reasoning, where arguments support conclusions. The reasoning in argumentation theory is not explained in terms of the interpretation of a defeasible condition, but in terms of the interactions between conflicting arguments. Surveys of this research field are [20, 7].

Although several approaches have been proposed for argument theory, Dung's approach presented in [11], is a unifying framework which has played an influential role on argumentation research and AI. In fact, Dung's approach has been influencing subsequent proposals for argumentation systems, *e.g.*, [4]. Besides, Dung's approach is mainly relevant in fields where conflict management plays a central role. For instance, Dung showed that his theory naturally captures the solutions of the theory of n-person game and the well-known stable marriage problem. In our case, the motivation of this research is to support medical decision-making in multi-agent systems [22, 13].

Dung's framework is captured by four argumentation semantics: *stable semantics*, *preferred semantics*, *grounded semantics*, and *complete semantics*. The central notion of these semantics is the *acceptability of the arguments*. An argument is called *acceptable* if and only if it belongs to a set of arguments which is called *extension*.

Although Dung's framework is captured by four argumentation semantics, the main semantics for collective acceptability are the grounded semantics and the preferred semantics. The first one adheres to the so-called unique-status approach, since for a given argumentation framework it always identifies a single extension, called grounded extension. The preferred semantics follows, instead, a multiple-status approach by identifying a set of preferred extensions. Since, the preferred semantics overcomes the limitations of the stable semantics and the grounded semantics, it is regarded as the most satisfactory approach. For instance, John

Pollock made preferred semantics one of the key ingredients of his revised formalism [19]. Also, it has been shown that some non-monotonic logic programming semantics can be viewed as a special form of this abstract argumentation semantics [5, 11].

Nowadays, it has been pointed out that the preferred semantics has some problems [20, 3, 6]. Although, the preferred semantics' problems happen in some particular cases, it is not difficult to build examples in our medical domain where preferred semantics' problems protrude [13]. So, we need to find some new abstract argumentation semantics, however it is clear that the new argumentation semantics mostly follows the preferred semantics' philosophy, since preferred semantics has shown its utility.

In this paper, we introduce an interesting and cautious preferred semantics' extension which is called preferred$^+$ semantics. The preferred$^+$ semantics has some interesting properties. For instance, one of the computational problems of the preferred semantics is that there are few algorithms to infer preferred extensions. As, the preferred$^+$ semantics is characterized by a proper representation of an argumentation framework and minimal models. Then, it is possible to use several techniques in order to find preferred$^+$ extensions. For instance, by assuming a polynomial time prepossessing of an argumentation framework, we can use the following techniques: DLV System[10, 15, 14, 1], UNSAT- algorithms[1], SAT-solvers[1], Graph theory [8][1].

It is well-known that the decision problem of the preferred extensions of an argumentation framework is CO-NP-Complete [12]. Hence, it is important to explore alternative approaches to solve the problem. In that sense, we propose to use a novel graph algorithm called *SI-COBRA* [17] to infer preferred$^+$ extensions. Notice that, the preferred$^+$ semantics is an extension of the preferred semantics, so this technique is also a new technique to infer preferred extensions.

The SI-COBRA algorithm is capable to detect instances of a graph G' in a graph G (subgraph isomorphism detection) using a linear sequence of codes to represent the graphs. As a result, the algorithm might show the association between the vertices and edges of G' with the corresponding vertices and edges of G (when a valid mapping exists). This algorithm has successfully been applied in theoretical and practical domains. For example, it has been used in genome databases in order to find low-complexity sequences [18], in web-log files to find access patterns [16] and in chemical compound domains [16].

The rest of the paper is structured as follows: In §2, we present the syntax and semantics of our logic programs, and also a short description of Dung's approach. In §3, we do a short road map of the basic principles which any preferred semantics' extension should always satisfy. In §4, we present our novel semantics. In §5, we present how to use the SI-COBRA algorithm for inferring preferred$^+$ extensions. And finally in the last section, we present our conclusions.

2. Background

In this section, we define the syntax of the logic programs that we will be using in this paper. Also, we will present a short description of the Dung's argumentation theory.

2.1. Syntax

A literal is an atom, a, or the negation of an atom $\neg a$. Given a set of atoms $\{a_1, ..., a_n\}$, we write $\neg\{a_1, ..., a_n\}$ to denote the set of atoms $\{\neg a_1, ..., \neg a_n\}$. A normal clause, C, is denoted: $a \leftarrow l_1, \ldots, l_n$, where $n \geq 0$, a is an atom, and each l_i is a literal. When $n = 0$ the clause is an abbreviation of $a \leftarrow \top$[2], where $\top$ is $\neg\bot$ (tratar de que el superindice de la nota no aparente ser un exponente que eleva la base al cuadrado). Sometimes, we denote a clause C by $a \leftarrow \mathcal{B}^+,\ \neg\mathcal{B}^-$, where $\mathcal{B}^+$ contains all the positive body atoms and $\mathcal{B}^-$ contains all the negative body atoms. We also use $body(C)$ to denote $\mathcal{B}^+ \cup \neg\mathcal{B}^-$. A normal program P is a finite set of normal clauses, formally a normal program is a conjunction of its normal clauses. We denote by $HEAD(P)$ the set $\{a | a \leftarrow \mathcal{B}^+,\ \neg\mathcal{B}^- \in P\}$. A signature $\mathcal{L}$ is a finite set of elements that we call atoms. We denote by $\mathcal{L}_P$ the signature of P, *i.e.* the set of atoms that occur in P. Given a signature $\mathcal{L}$, we write $Prog_{\mathcal{L}}$ to denote the set of all programs defined over $\mathcal{L}$. We point out that whenever we consider logic programs, our negation $\neg$ corresponds to the default negation *not* used in Logic Programming.

2.2. Semantics

Let P be a normal program. An interpretation I is a mapping from $\mathcal{L}_P$ to $\{0, 1\}$, where the generalization of I to connectives is as follows: $I(a \wedge b) = min\{I(a), I(b)\}$, $I(a \vee b) = max\{I(a), I(b)\}$, $I(a \leftarrow b) = 0$ if and only if $I(b) = 1$ and $I(a) = 0$, $I(\neg a) = 1 - I(a)$, $I(\bot) = 0$. An

1 It is important, to point out that all these techniques are also useful to infer only the preferred semantics. This because the preferred extensions are also characterized by minimal models (see [14] for details).

2 or simply a.

interpretation M is called a model of P if and only if for each clause $c \in P$, $M(c) = 1$. Finally, M is a minimal model of P if it does not exist a model M' of P such that $M' \subset M$.

Our novel argumentation semantics is based on rewriting systems, so we define some transformation rules for logic programs.

Definition 1 (Basic Transformation Rules) *[9] A transformation rule is a binary relation on $Prog_{\mathcal{L}}$. The following transformation rules are called* basic. *Let a program $P \in Prog_{\mathcal{L}}$ be given.*

RED$^+$: *This transformation can be applied to P, if there is an atom a which does not occur in HEAD(P). RED$^+$ transforms P to the program where all occurrences of $\neg a$ are removed.*

RED$^-$: *This transformation can be applied to P, if there is a rule $a \leftarrow \top \in P$. RED$^-$ transforms P to the program where all clauses that contain $\neg a$ in their bodies are deleted.*

Success: *Suppose that P includes a fact $a \leftarrow \top$ and a clause $q \leftarrow body$ such that $a \in body$. Then we replace the clause $q \leftarrow body$ by $q \leftarrow body \setminus \{a\}$.*

Failure: *Suppose that P contains a clause $q \leftarrow body$ such that $a \in body$ and $a \notin HEAD(P)$. Then, we erase the given clause.*

Loop: *We say that P_2 results from P_1 by* Loop_A *if, by definition, there is a set A of atoms such that*

1. *for each rule $a \leftarrow body \in P_1$, if $a \in A$, then $body \cap A \neq \emptyset$,*
2. *$P_2 := \{a \leftarrow body \in P_1 : body \cap A = \emptyset\}$,*
3. *$P_1 \neq P_2$.*

Essentially, any basic transformation rule reduces a normal program P based on the syntactic information that there is in P. In this paper, we denote by CS_0 the set of transformation rules of Definition 1. When, we apply a set of transformation rules *e.g.*, CS_0, to a normal program P, there is a reduced program P' where none of the transformation rules can be applied to P'. Usually, P' is called the uniquely determined normal form of the program P *w.r.t.* a set of transformation rules *e.g.*, CS_0. We shall denote the uniquely determined normal form of a program P *w.r.t.* CS_0 by $norm_{CS_0}(P)$.

In order to illustrate the basic transformation rules, let us consider the following example.

Example 1 *Let P be the following normal program:*

$$d(b) \leftarrow \neg d(a). \qquad d(c) \leftarrow \neg d(b).$$
$$d(c) \leftarrow d(a).$$

*Now, let us apply CS_0 to P. Since $d(a) \notin HEAD(P)$, then, we can apply **RED$^+$** to P. Thus we get:*

$$d(b) \leftarrow \top. \qquad d(c) \leftarrow \neg d(b).$$
$$d(c) \leftarrow d(a).$$

*Notice that now we can apply **RED$^-$** to the new program, thus we get:*

$$d(b) \leftarrow \top. \qquad d(c) \leftarrow d(a).$$

*Finally, we can apply **Failure** to the new program, thus we get:*

$$d(b) \leftarrow \top.$$

This last program is called the normal form of P w.r.t. CS_0, because none of the transformation rules from CS_0 can be applied.

2.3. Background: Argumentation

The fundamental Dung's definition is the concept called argumentation framework which is defined as follows:

Definition 2 *[11] An argumentation framework is a pair $AF = \langle AR, attacks \rangle$, where* AR *is a finite set of arguments, and* attacks *is a binary relation on* AR, i.e. attacks $\subseteq AR \times AR$.

Figure 1. A single argumentation framework

Any argumentation framework could be regarded as a directed graph. For instance, if $AF = \langle \{a, b, c\}, \{(a, b), (b, c)\} \rangle$, then AF is represented as in Figure 1. We say that *a attacks b* (or *b* is attacked by *a*) if $attacks(a, b)$ holds. Similarly, we say that a set S of arguments attacks b (or b is attacked by S) if b is attacked by an argument in S. For instance in Figure 1, $\{a\}$ attacks b.

Definition 3 [11] *A set S of arguments is said to be conflict-free if there are no arguments A, B in S such that A attacks B.*

For instance, the sets $\{a\}$, $\{b\}$, and $\{a, c\}$ are conflict-free sets *w.r.t.* Figure 1.

Definition 4 [11] *(1) An argument $A \in AR$ is acceptable with respect to a set S of arguments iff for each argument $B \in AR$: If B attacks A then B is attacked by S. (2) A conflict-free set of arguments S is admissible iff each argument in S is acceptable w.r.t. S.*

One of the semantics of the Dung's approach which has played an influential role on argumentation research is the preferred semantics. This semantics is defined as follows:

Definition 5 *[11] A preferred extension of an argumentation framework AF is a maximal (w.r.t. inclusion) admissible set of AF.*

The admissible sets of Figure 1 are $\{a\}$ and $\{a, c\}$, then the only preferred extension is $\{a, c\}$. Dung suggested other argumentation semantics, however we do not present them here, because they are not the motivation of this paper.

Dung [11] defined some important concepts *w.r.t.* the relationship between arguments when they are taking part of a sequence of attacks.

- An argument B *indirectly attacks* A if there exists a finite sequence $A_0, \ldots, A_{2n+1}$ such that 1) $A = A_0$ and $B = A_{2n+1}$, and 2) for each i, $0 \leq i \leq 2n$, A_{i+1} attacks A_i.
- An argument B *indirectly defends* A if there exists a finite sequence $A_0, \ldots, A_{2n}$ such that 1) $A = A_0$ and $B = A_{2n}$ and 2) for each i, $0 \leq i \leq 2n$, A_{i+1} attacks A_i.
- An argument B is said to be *controversial w.r.t.* A if B indirectly attacks A and indirectly defeats A.
- An argument is *controversial* if it is controversial *w.r.t.* some argument A.

Based on these concepts, Dung defined a family of argumentation frameworks.

Definition 6 *[11] An argumentation framework is limited controversial if there exists no infinite sequence of arguments $A_0, \ldots, A_n, \ldots$ such that A_{i+i} is controversial w.r.t. A_i.*

We will consider special kind of arguments for defining our new semantics. These arguments are called acyclic arguments, defined as follows:

Definition 7 *[14] Let $AF = \langle AR, Attacks \rangle$ be an argumentation framework and $A \in AR$. A is an acyclic argument if there is not a sequence of attacks $A_0, \ldots, A_n$ such that 1) $A = A_0$ and $A = A_n$, and 2) for each i, $0 \leq i \leq n-1$, A_{i+1} attacks A_i.*

3. Looking for a new argumentation-based semantics

As we commented in the introduction, it has been found that there are some problems in the behaviour of the prefered semantics [20, 3, 6]. Hance, we need to find some new abstract argumentation semantics, however the new argumentation semantics most it follows preferred semantics' philosophy, since preferred semantics has been shown its utility. Then, one of the greatest steps is to find some fundamental principles for argumentation semantics. The intension of this short section is to do a short road map of the basic principles which any preferred semantics' extension should always satisfy.

There are two main principles which are well accepted [20, 2]: **Conflict-free.** This principle suggests that any set of acceptable arguments (extension) is be a conflict-free set. **Reinstatement.** In the case of an argumentation framework consisting of a defeat chain (see Figure 1) it is widely acceptable that the initial node, which has no defeaters *e.g.*, argument a, and all the other nodes in odd position, whose defeaters are defeated by an undefeated node *e.g.*, argument c, should be regarded as undefeated, while nodes in even position *e.g.*, argument b, should be regarded as defeated.

To be close to preferred semantics' philosophy, it is relevant to identify families of argumentation frameworks where the preferred semantics' behavior is well accepted. For instance, an interesting family of argumentation frameworks is the limited controversial argumentation framework. The limited controversial frameworks have the property of begin *coherent*. This means that the stable and preferred semantics coincide. And also, any limited controversial framework has at least a nonempty complete extension [11]. This suggests a principle that we called **Coherent**: For any limited controversial argumentation framework, a new argumentation semantics should coincide with the preferred semantics.

Admissibility is the basic notion of acceptability for Dung's approach. It relies upon the notion of defence and is such that if an argument does not belong to an admissible set, then its attacks cannot be used to defend any argument of the admissible set. This restriction close to the preferred semantics looks only for sets of arguments which are defended by themselves. So, it is clear that any preferred semantics' extension shall relax the admissible sets' definition. However that relaxation most include the admissible sets. We called this restrictions **Preserving admissibility**.

4. Preferred$^+$ semantics

In this section, we present an interesting extension of the preferred semantics which is based on a proper representation of an argumentation framework, in terms of normal programs, and minimal models.

The representation of an argumentation framework that we are going to consider was first introduced in [14] and it is based on normal programs. We use the predicate *d(X)*, the intended meaning of *d(X)* is "X is defeated".

To build our representation, we consider the set of arguments which attack a particular argument.

Definition 8 *Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework. The direct defeaters of $A \in AR$ is the set $D(A) := \{B | (B, A) \in Attacks\}$.*

The representation of any argument, in terms of normal clauses, is defined as follows:

Definition 9 *[14] Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Phi(A)$ as follows:*

If $|D(A)| = 1$ and A is an acyclic argument

$$\Phi(A) := d(A) \leftarrow \neg d(B) \text{ where } B \in D(A)$$

otherwise

$$\Phi(A) := (\bigwedge_{B \in D(A)} d(A) \leftarrow \neg d(B)) \wedge (\bigwedge_{B \in D(A)} d(A) \leftarrow \bigwedge_{C \in D(B)} d(C))$$

So, the representation of an argumentation framework is defined as follows.

Definition 10 *[14] Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework. We define its associated normal program as follows:*

$$\Phi_{AF} := \bigwedge_{A \in AR} \Phi(A)$$

In order to illustrate the definitions, let us consider the following example.

Example 2 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, where $AR := \{a, b, c, d\}$ and $attacks := \{(a, b), (b, a), (a, c), (b, c), (c, d)\}$ (see Figure 2). Then, Φ_{AF} is :*

$$\begin{array}{ll} d(a) \leftarrow \neg d(b). & d(a) \leftarrow d(a). \\ d(b) \leftarrow \neg d(a). & d(b) \leftarrow d(b). \\ d(c) \leftarrow \neg d(b). & d(c) \leftarrow d(b). \\ d(c) \leftarrow \neg d(a). & d(c) \leftarrow d(a). \\ d(d) \leftarrow \neg d(c). & \end{array}$$

In order to define our new semantics, we define the function $s(E)$.

Definition 11 *Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework. Given a set of arguments $E \subseteq AR$, s(E) is defined as follows: $s(E) := \{d(a) | a \in AR \setminus E\}$*

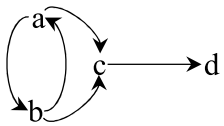

Figure 2. An argumentation framework with a two-length cycle.

Essentially, we understand that if E is a set of acceptable arguments then $s(E)$ will be the set of defeated arguments. In other words, $s(E)$ expresses the complement of the set E *w.r.t.* AR. Now, we define our concept of admissible set.

Definition 12 *Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework and $E \subseteq AR$. We say that E is a admissible$^+$ set of AF if and only if $s(E)$ is model of $norm_{CS_0}(\Phi_{AF})$.*

In order to illustrate the definition, let us consider the program Φ_{AF} from Example 2. We can see that $norm_{CS_0}(\Phi_{AF}) = \Phi_{AF}$, because we can not apply any transformation $R \in CS_0$ to Φ_{AF}. So, we can consider Φ_{AF}'s models in order to find the admissible set of AF. Φ_{AF} has two modes: $\{d(a), d(c)\}$ and $\{d(b), d(c)\}$. Then, the admissible$^+$ sets of AF are: $\{a, d\}$ and $\{c, d\}$. An important property of our admissible$^+$ sets is that they are conflict-free sets.

Proposition 1 *Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework and $E \subseteq AR$. E is an admissible$^+$ set of AF then E is a conflict-free set of arguments*

Now, by considering the mapping Φ, we define our extension of the preferred semantics. The definition of our new semantics is defined in terms of minimal models as follows.

Definition 13 *Let $AF = \langle AR, attacks\rangle$ be an argumentation framework, $E \subseteq AR$. E is a preferred$^+$ extension of AF if and only if s(E) is a minimal model of $norm_{CS_0}(\Phi_{AF})$.*

As, we saw the argumentation framework of Example 2 has two models, in that case both models are minimal, then both models correspond to the *preferred$^+$ extensions* which are : $\{a, d\}$ and $\{c, d\}$. Particulary, in Example 2 both preferred semantics and preferred$^+$ semantics coincide. In fact, the preferred$^+$ semantics and preferred semantics coincide for any limited controversial argumentation framework.

Proposition 2 *Let $AF := \langle AR, attacks\rangle$ be a limited controversial argumentation framework. Then, each preferred extension is a preferred$^+$ extension and vice versa.*

With the following theorem, we formalize that the preferred$^+$ semantics is an extension of the preferred semantics.

Theorem 1 *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. If E is a preferred extension of AF, then there is a preferred$^+$ extension E^+ such that $E \subseteq E^+$.*

Now, let us consider another example where the preferred$^+$ semantics takes relevance.

Example 3 *Let $AF = \langle AR, attacks\rangle$ be an argumentation framework, where $AR := \{a, b, c, d, e\}$ and attacks $:= \{(a, c), (c, b), (b, a), (a, d), (b, d), (c, d), (d, e)\}$ (see Figure 3). Structurally the only difference between the argumentation framework of Example 2 and AF is that one has a two-length cycle and the another one has a tree-length cycle. As we mentioned before, the preferred extensions of Example 2 are: $\{d(a), d(c)\}$ and $\{d(b), d(c)\}$. Notice that the intersection of both preferred extensions is $\{c\}$. This means that we can always consider the argument c as acceptable. Now, by applying the preferred semantics to AF, the only preferred extension is : $\{\}$. This means that length-cycle affects to the preferred semantics, because we can expect to get at least the argument e as an accepted argument. In fact, this is a widely discussed example [20, 3]. Now, let us consider the preferred$^+$ semantics. The only preferred$^+$ extension is: $\{e\}$.*

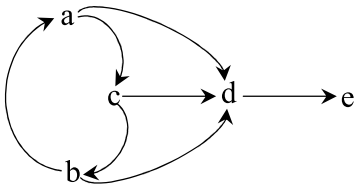

Figure 3. An argumentation framework with a three-length cycle.

5. Computing the preferred$^+$ semantics

In this section, we present a technique to infer preferred$^+$ extensions. Since, the preferred$^+$ semantics is an extension of the preferred semantics, this technique is also a new form to infer preferred extensions.

5.1. A Graph-Based Representation

In this subsection, we describe the way in which a normal program Φ_{AF} can be described through a graph-based representation. The idea consists on building a *subgraph isomorphism* instance (G', G), where G' is a complete graph that represents a possible model of Φ_{AF}, and G represents the program Φ_{AF}.

We start by presenting the definitions of graph, subgraph and isomorphism. These definitions are oriented to work with undirected labeled graphs. After that, the proposed reduction/representation is presented.

Definition 14 *An undirected graph is a 6-tuple $G = (V, E, L_V, L_E, \alpha, \beta)$, where:*

- *$V = \{v_i | i = 1, \ldots, m\}$, is the finite set of vertices, $V \neq \emptyset$* [3]
- *$E \subseteq \{\{v_i, v_j\} : v_i, v_j \in V\}$*
- *L_V, is a set of vertex labels*
- *L_E, is a set of edge labels*
- *$\alpha : V \rightarrow L_V$, is a function assigning labels to the vertices*
- *$\beta : E \rightarrow L_E$, is a function assigning labels to the edges*

In our approach, we use vertices labels to store the information of a program. This is the main reason of why we work with undirected labeled graphs.

Definition 15 *Let G be a graph, where $G = (V, E, L_V, L_E, \alpha, \beta)$. A* **subgraph** *$G^S$ of G, denoted by $G^S \subseteq G$, $G^S = (V^S, E^S, L_V^S, L_E^S, \alpha^S, \beta^S)$ is a graph such that $V^S \subseteq V$, $E^S \subseteq E$, $\alpha^S \subseteq \alpha$ and $\beta^S \subseteq \beta$.*

Definition 16 *Given two graphs G' and G, G' is isomorphic to G, denoted by $G' \cong G$, if there exists $f : V' \rightarrow V$ and $g : E' \rightarrow E$ as bijections, where:*

- *$\forall v' \in V', \alpha'(v') = \alpha(f(v'))$*
- *$\forall \{v'_i, v'_j\} \in E', \beta'(\{v'_i, v'_j\}) = \beta(g(\{v'_i, v'_j\}))$*

In other words, an isomorphism between G' and G exists if the topology and labeling of both graphs is exactly the same.

Definition 17 *Let G' and G be two graphs. G' is a* **subgraph isomorphic** *to G if there exists $G^S \subseteq G$ such that $G' \cong G^S$.*

Now, we describe the process that transforms a normal program Φ_{AF} into its graph-based representation. This process is performed in three steps:

3 In this document, subindex i is a unique value associated to each vertex and it is only used to identify each vertex.

1. First, we get the normal form of the program Φ_{AF}, denoted by Φ^*_{AF}, this means that $\Phi^*_{AF} = norm_{CS_0}(\Phi_{AF})$;
2. Second, Φ^*_{AF} is transformed into its conjunctive normal form, represented by Φ'_{AF};
3. And finally, Φ'_{AF} **is represented through a graph**.

Clearly, the first two steps are easy to perform. In this section, we are interested on describing a process to transform Φ'_{AF} into its graph-based representation and finding Φ_{AF}'s models through the *subgraph isomorphism problem*. In the rest of this subsection, we describe the process to transforming Φ'_{AF} to its graph-based representation.

Let $\hat{C} = \{C_1, \ldots, C_n\}$ be a set of clauses, $n = |\Phi'_{AF}|$, where $C_x \in \hat{C}$ represents a conjuntive normal clause from Φ'_{AF}. Each C_x is denoted by $C_x = \{u_1, \ldots, u_{s_x}\}$, $s_x = |C_x|$, where $u_i \in C_x$ is either $\neg d(A)$ or $d(A)$, $A \in AR$ (for the sake of simplicity, we refer to u_i as an atom).

It is important to remark that a set of clauses $\hat{C} = \{C_1, \ldots, C_n\}$ is satisfiable iff: $\forall C_x \in \hat{C} : \exists u \in C_x, u$ is true. The next definition presents the way in which a set $\hat{C}$ of clauses can be represented by a graph. Let $f(x, u)$ be a function that maps an atom u from C_x to a vertex in the graph.

Definition 18 *A graph-based representation of a set $\hat{C} = \{C_1, \ldots, C_n\}$ of clauses is the graph $\hat{G} = (V, E, L_V, L_E, \alpha, \beta)$, where:*

- $V = \bigcup_{\forall C_x \in \hat{C}} V_x$. *If $f(x, u)$ maps each atom u from each clause C_x to a vertex v, then $V_x = \{v : f(x, u) = v, \forall u \in C_x\}$.*
- $\{v_i, v_j\} \in E$ *if holds:*
  a) $v_i, v_j \in V$ and,
  b) if $v_i \in V_x$, then $v_j \notin V_x$ and,
  c) if $f(x, u) = v_i$, $f(y, w) = v_j$ and u, w are the same atom, then $u \neq \neg w$ or $\neg u \neq w$.
- $L_V = \{1, 2, 3, \ldots, m\}$, $|L_E| = 1$ *(all the edges have the same label)*
- $\alpha : V \to L_V$, *where: $\alpha(v_i) = x$ if $f(x, u) = v_i$*
- $\beta : E \to L_E$

Note that each atom in each clause $C_x \in \hat{C}$ is mapped to one vertex in $\hat{G}$. Also, each clause C_x derives a set of vertices V_x, where each vertex $v \in V_x$ has the label x and V is conformed from the union of the sets V_x. On the other hand, there are not edges belonging to E that come from the same V_x and, there are not edges between vertices that represent the same atom (u) and one of them is negated.

It is interesting to remark that we assign the same label to each member of a set of vertices V_x. With this proposal, our objective is to reduce the number of operations to perform in the association process (at the moment of finding the subgraph isomorphic), because if a vertex v in V_x has already been considered in the mapping process, then none of the remaining vertices in V_x will be considered again.

Based on Definition 18, we introduce the following lemma.

Lemma 1 *Let Φ^*_{AF} be a normal program and $\hat{C}$ represents its conjuntive normal clause. A graph-based representation of Φ^*_{AF} is a graph $\hat{G}$ such that vertices in $\hat{G}$ represents the literals of each clause $C_x \in \hat{C}$ and edges in $\hat{G}$ represent the valid associations between atoms that belong to different clauses $C_x \in \hat{C}$.*

By considering Lemma 1, we will show how a normal program Φ_{AF} can be reduced to an instance $(G', \hat{G})$ of the subgraph isomorphism problem.

First, we define graph $\hat{G}$ as in Definition 18 and $G' = (V', E', L'_V, L'_E, \alpha', \beta')$, where: $V' = \{v'_i : i = 1, \ldots, n\}$ (n is the number of clauses), $E' = \{\{v'_i, v'_j\} : i \neq j\}$, $L'_V = L_V$, $L'_E = L_E$, $\alpha' : V' \to L'_V$ where $\alpha(v_i) = i$ and $\beta' : E' \to L'_E$.

Note that G' is a complete graph with n vertices, because we need to find a subgraph where there should exist an edge between each pair of vertices and a vertex representing one atom for each clause (this is the reason of why G' has n vertices). Therefore, G' represents a possible model of Φ'_{AF}. Then, we build an instance $(G', \hat{G})$ of the subgraph isomorphism problem where we want to find if $\exists S : S \subseteq \hat{G}$ and $S \cong G'$. If $(G', \hat{G})$ is a yes-instance of the subgraph isomorphism problem, then S satisfies $\hat{C}$ through the atoms represented by the vertices in G'.

Example 4 *Let Φ_{AF} be the normal program from Example 2. Then, its corresponding Φ'_{AF} is: $d(a) \vee d(b)$, $d(c) \vee d(b)$, $d(c) \vee d(a)$, $d(d) \vee d(c)$, $d(c) \vee \neg d(b)$, $d(c) \vee \neg d(a)$.*
So, the corresponding $\hat{C}$ of Φ'_{AF} is $\hat{C} = \{C_1, \ldots, C_6\}$ where: $C_1 = \{d(a), d(b)\}$, $C_2 = \{d(c), d(b)\}$, $C_3 = \{d(c), d(a)\}$, $C_4 = \{d(d), d(c)\}$, $C_5 = \{d(c), \neg d(b)\}$, $C_6 = \{d(c), \neg d(a)\}$. Based on the above mentioned, the corresponding function $f(x, u)$ of $\hat{C}$ is defined as follow:

$$
\begin{array}{ll}
f(1, d(a)) = v_1. & f(4, d(d)) = v_7 \\
f(1, d(b)) = v_2. & f(4, d(c)) = v_8 \\
f(2, d(c)) = v_3. & f(5, d(c)) = v_9 \\
f(2, d(b)) = v_4. & f(5, \neg d(b)) = v_{10} \\
f(3, d(c)) = v_5. & f(6, d(c)) = v_{11} \\
f(3, d(a)) = v_6. & f(6, \neg d(a)) = v_{12}
\end{array}
$$

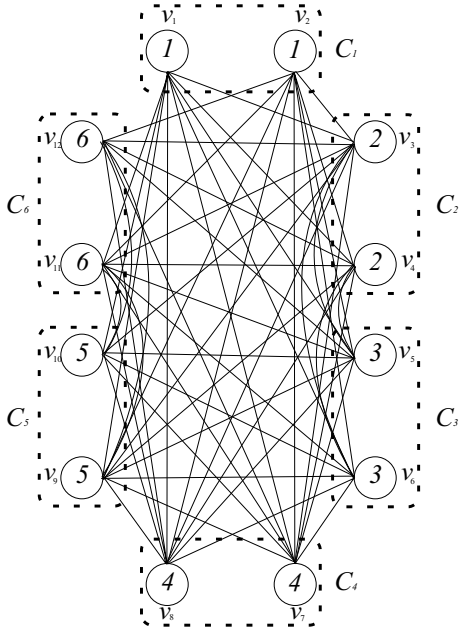

Figure 4. The Graph-Based Representation of the normal program Φ_{AF}, example 2

Bearing in mind the above mentioned, we define a graph $\hat{G} = (V, E, L_V, L_E, \alpha, \beta)$, where:

- $V = \{V_1, \ldots, V_6\}$, $V_1 = \{v_1, v_2\}$, $V_2 = \{v_3, v_4\}$, $V_3 = \{v_5, v_6\}$, $V_4 = \{v_7, v_8\}$, $V_5 = \{v_9, v_{10}\}$ *and* $V_6 = \{v_{11}, v_{12}\}$
- $E = \{\{v_1, v_3\}, \{v_1, v_4\}, \{v_1, v_5\}, \{v_1, v_3\}, \{v_1, v_6\}, \{v_1, v_7\}, \{v_1, v_8\}, \{v_1, v_9\}, \{v_1, v_{10}\}, \{v_1, v_{11}\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_2, v_5\}, \{v_2, v_6\}, \{v_2, v_7\}, \{v_2, v_8\}, \{v_2, v_9\}, \{v_2, v_{11}\}, \{v_2, v_{12}\}, \{v_3, v_5\}, \{v_3, v_6\}, \{v_3, v_7\}, \{v_3, v_8\}, \{v_3, v_9\}, \{v_3, v_{10}\}, \{v_3, v_{11}\}, \{v_3, v_{12}\}, \{v_4, v_5\}, \{v_4, v_6\}, \{v_4, v_7\}, \{v_4, v_8\}, \{v_4, v_9\}, \{v_4, v_{11}\}, \{v_4, v_{12}\}, \{v_5, v_7\}, \{v_5, v_8\}, \{v_5, v_9\}, \{v_5, v_{10}\}, \{v_5, v_{11}\}, \{v_5, v_{12}\}, \{v_6, v_7\}, \{v_6, v_8\}, \{v_6, v_9\}, \{v_6, v_{10}\}, \{v_6, v_{11}\}, \{v_7, v_9\}, \{v_7, v_{10}\}, \{v_7, v_{11}\}, \{v_7, v_{12}\}, \{v_8, v_9\}, \{v_8, v_{10}\}, \{v_8, v_{11}\}, \{v_8, v_{12}\}, \{v_9, v_{11}\}, \{v_9, v_{12}\}, \{v_{10}, v_{11}\}, \{v_{10}, v_{12}\}\}$
- $L_V = \{1, 2, 3, 4, 5, 6\}$
- $L_E = \{l\}$
- $\alpha : V \rightarrow L_V$
- $\beta : E \rightarrow L_E$

Note that there is not edge between v_1 and v_{12} because $f(1, d(a)) = v_1$ and $f(6, \neg d(a)) = v_{12}$ have the same atom $d(a)$, but one of them is negated. A similar argumentation is applicable to v_6 and v_{12}, v_2 and v_{10}, v_4 and v_{10}. Figure 4 shows the graph $\hat{G}$ of example 4.

G'	G	G'	G	G'	G
(v1,v2)	(v1,v3)	(v4,v2)	(v7,v3)	(v5,v6)	(v9,v11)
(v2,v3)	(v3,v5)	(v4,v5)	(v7,v9)	(v6,v1)	(v11,v1)
(v3,v1)	(v5,v1)	(v5,v1)	(v9,v1)	(v6,v2)	(v11,v3)
(v3,v4)	(v5,v7)	(v5,v2)	(v9,v3)	(v6,v3)	(v11,v5)
(v4,v1)	(v7,v1)	(v5,v3)	(v9,v5)	(v6,v4)	(v11,v7)

Figure 5. SI-COBRA output example

Finally, it is necessary to define a graph G' for building an instance $(G', \hat{G})$. Since $|\hat{C}| = 6$, then $G' = (V', E', L'_V, L'_E, \alpha', \beta')$, where:

- $V = \{v_1, \ldots, v_6\}$
- $E' = \{\{v'_i, v'_j\} : i \neq j\}$
- $L'_V = \{1, 2, 3, 4, 5, 6\}$
- $L'_E = \{l\}$
- $\alpha' = \{\alpha'(v_1) = 1, \alpha'(v_2) = 2, \alpha'(v_3) = 3, \alpha'(v_4) = 4, \alpha'(v_5) = 5, \alpha'(v_6) = 6\}$
- $\beta' : E' \rightarrow L'_E$

In the next subsection, we expose the way in which an instance $(G', \hat{G})$ can be solved using the SI-COBRA algorithm.

5.2. Computing the preferred$^+$ semantics with SI-COBRA

As we mentioned earlier, with SI-COBRA we have the option of finding only a valid mapping (and stop the process) or, find all possible mappings (if there are more than one). So, given an argumentation framework AF and by considering the graph-representation of the program Φ_{AF}. We can solve three different queries: 1.- find an admissible$^+$ set of AF, this means to find a model of the program $norm_{CS_0}(\Phi_{AF})$ (see Definition 12); 2.- find all the admissible$^+$ sets of AF; and 3.- find a preferred$^+$ extension which contains a particular argument, this means to find a particular minimal model of $norm_{CS_0}(\Phi_{AF})$ (see Definition 13).

As an example, let us consider the input graphs G' and $\hat{G}$ from Example 4. After running an implementation of the SI-COBRA algorithm with this input, we get 24 different matches between vertices and edges from G' and G. Figure 5 shows a SI-COBRA's output format example, where mappings are depicted based on edges and their corresponding vertices.

Based on function $f(u, x)$ (see Definition 18), we can retrieve the association between vertices and atomss of G and Φ^*_{AF} respectively. In our example, the literals that represent a solution are: 1.- $\{d(a), d(c), d(d)\}$, this means that $\{b\}$ is an admissible$^+$ set of AF; however $\{b\}$ is not a preferred$^+$ extension because

$\{d(a), d(c), d(d)\}$ is not a minimal model of Φ_{AF}; 2.- $\{d(a), d(c)\}$, this means that $\{b, d\}$ is an admissible set of AF, in fact it is a preferred$^+$ extension because $\{d(a), d(c)\}$ is a minimal models of Φ_{AF}.

Although, SI-COBRA does not compute directly the preferred$^+$ extensions, because SI-COBRA only compute models of Φ_{AF}. There are two ways that allow us to find it: analyze all results or, use an algorithm that generates the preferred$^+$ extensions in a recursive way. If we analyze all results, we need to find a model with the smaller number of atoms. On the other hand, it is possible to use the following algorithm to find the preferred$^+$ semantics [1]:

1. Let Φ^* be the input normal program and U the set of atoms over Φ^*
2. Compute a model of Φ^*, called M
3. If there are not models, then Φ^* is inconsistent
4. Compute $\Phi^{**} = \Phi^* \cup \neg(U \backslash M) \cup \{\neg(u_1), \ldots, \neg(u_n)\}$, where $u_1, \ldots, u_n$ are elements of M
5. Compute a model of Φ^{**}, called M'
6. If there are not models of Φ^{**}, then M is minimal. Finish
7. In other case, $M := M', \Phi^* := \Phi^{**}$. Return to 4

Finally, if we require that an atom (or set of atoms) appears as part of our results, then we can do it in the following way. First, consider that M is a model of Φ^*, where an atom $l \in M$ and $\hat{C} = C_1, \ldots, C_n$ is the set of clauses associated to Φ^*. Then, if we define $\hat{C}' = \hat{C} \cup C_+$, where $C_+ = \{l\}$, then all possible models of Φ^* include l as a part of their solutions. Since SI-COBRA finds all possible solutions, then we will also find M with $\hat{C}'$. With this argument, we show the way in which we can find models where it is necessary to consider a specific literal.

6. Conclusions

We presented a novel argumentation semantics called preferred$^+$ semantics which is characterized by normal programs and minimal models. Preferred$^+$ semantics semantics follows preferred$^+$ semantics' philosophy that is one of the most successful Dung's semantics. As, Preferred$^+$ semantics is characterized by minimal models there is a wide variety of techniques in order to infer preferred$^+$ extensions. In this paper, we explore the use of a novel graph algorithm called SI-COBRA which has successfully been applied in theoretical and practical domains, in order to infer preferred$^+$ extensions. Since, the preferred semantics was characterized by minimal model in [14] this technique is also a novel technique to infer the preferred semantics.

Although the preferred$^+$ semantics is defined for Dung's argumentation framework, it is also applied to value-based argumentation frameworks [4]. This is because, according to [14], the decision problem of the preferred extensions of a value-based argumentation framework is polynomial time reducible to the decision problem of the preferred extensions of an argumentation framework.

Acknowledgements

J.C. Nieves thanks to CONACyT for his Doctorate Grant. J.C. Nieves and U. Cortés were partially supported by the grant FP6-IST-002307 (ASPIC). The views expressed in this paper are not necessarily those of ASPIC consortium.

References

[1] O. Arieli, M. Denecker, B. V. Nuffelen, and M. Bruynooghe. Database repair by signed formulae. In *FoIKS*, volume 2942 of *LNCS*, pages 14–30. Springer, 2004.

[2] P. Baroni and M. Giacomin. Evaluating argumentation semantics with respect to skepticism adequacy. In *ECSQARU 2005, Barcelona, Spain, July 6-8, 2005, Proceedings*, volume 3571 of *LNCS*, pages 329–340. Springer, 2005.

[3] P. Baroni, M. Giacomin, and G. Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168:162–210, October 2005.

[4] T. Bench-Capon. Value-based argumentation frameworks. In *Proceedings of Non Monotonic Reasoning*, pages 444–453, 2002.

[5] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93:63–101, 1997.

[6] M. Caminada. Contamination in formal argumentation systems. In *BNAIC 2005 - Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence, Brussels, Belgium, October 17-18*, pages 59–65, 2005.

[7] C. I. Chesñevar, A. G. Maguitman, and R. P. Loui. Logical models of argument. *ACM Comput. Surv.*, 32(4):337–383, 2000.

[8] Y. Dimopoulos and A. Torres. Graph theoretical structures in logic programs and default theories. *Theor. Comput. Sci.*, 170(1-2):209–244, 1996.

[9] J. Dix, M. Osorio, and C. Zepeda. A general theory of confluent rewriting systems for logic programming and its applications. *Ann. Pure Appl. Logic*, 108(1-3):153–188, 2001.

[10] S. DLV. . http://www.dbai.tuwien.ac.at/proj/dlv/.

[11] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[12] P. E. Dunne and T. J. M. Bench-Capon. Complexity in value-based argument systems. In *JELIA*, volume 3229 of *LNCS*, pages 360–371. Springer, 2004.

[13] J. C. Nieves. Ph. D. Proposal : Supporting decision making in organ transplanting using argumentation theory. Universitat Politècnica de Catalunya, Software Department, 2006.

[14] J. C. Nieves, M. Osorio, and U. Cortés. Modeling argumentation based semantics using non-monotonic reasoning. Technical report, Universitat Politècnica de Catalunya, Software Department, 2005.

[15] J. C. Nieves, M. Osorio, C. Zepeda, and U. Cortés. Inferring acceptable arguments with answer set programming. In *Sixth Mexican International Conference on Computer Science (ENC 2005)*, pages 198–205. IEEE Computer Science Press, September 2005.

[16] I. Olmos and J. A. González. Web site usage patterns discovery using a graph based representation. *IX Ibero-American WorkShop on Machine Learning for Scientific Data Analysis*, pages 345–354, 2004.

[17] I. Olmos, J. A. González, and M. Osorio. Mining common patterns on graphs. *International Conference on Computational Intelligence and Security*, vol 1, LNAI3802:41–48, 2005.

[18] I. Olmos, J. A. González, and M. Osorio. Inexact graph matching: A case of study. *To appear in Proceedings of the 19th International FLAIRS Conference*, 2006.

[19] J. L. Pollock. *Cognitive Carpentry: a blueprint for how to build a person.* The MIT Press, May 4, 1995.

[20] H. Prakken and G. A. W. Vreeswijk. Logics for defeasible argumentation. In D. Gabbay and F. Günthner, editors, *Handbook of Philosophical Logic*, volume 4, pages 219–318. Kluwer Academic Publishers, Dordrecht/Boston/London, second edition, 2002.

[21] I. Rahwad, S. D. Ramchurn, N. R. Jennings, P. Mcburney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation. *The Knowledge Engineering Review*, 18(4):343–375, 2004.

[22] P. Tolchinsky, U. Cortés, J. C. Nieves, A. López-Navidad, and F. Caballero. Using arguing agents to increase the human organ pool for transplantation. In *Proc. of the Third Workshop on Agents Applied in Health Care (IJCAI 2005)*, 2005.